```
#!/usr/bin/perl
################################################################################################
#
#
#       NAME:    xtra_tech_capture_pcap.pl
#
#       DESCRIPTION:
#
#       This utility will parse a Extratech OCD Packet Capture Tool file
#
#       REVISION HISTORY:
#       08/2012        Created
#
#       USAGE:
#          perl xtra_tech_capture_pcap.pl 'some OCD Packet capture txt filename'
#
#       MODULES:
#       uses    GD::Graph                - to create pngs of plotting info
#               Statistics::R            - to interface to R Statistics environment
#               Statitics::Descriptive   - for stddev() and mean() methods
#
#       TO DOs / Ideas:
#
################################################################################################
##
use strict;
use English;
use Time::Local;
use Statistics::R;
use Statistics::Descriptive;
use GD::Graph::linespoints();

    my  %stationsToCounts = (
        PrimaryHome              => "-322000_160000",
        PrimarySaha              => "-322001_160000",
        PrimaryPmTl              => "-447520_26706",
        PrimaryCmTl              => "-454320_26706",
        PrimaryLeak              => "-476300_26706",
        PrimaryVtip              => "-500727_177506",
        PrimaryStat              => "-624128_140672",
        PrimaryLearn             => "-751260_104629",
        PrimaryRightCoverClear   => "-798691_90000",
        PrimarySeal              => "-887465_174625",
        PrimaryQtip              => "-904854_188390",
        PrimaryDump              => "-1035858_201266",
        PrimaryWetInc            => "-1212045_162101",
        PrimaryExternal          => "UNKNOWN",
        SecondaryStat            => "-973260_40639",
        SecondarySaha0           => "-1030222_75554",
        SecondarySaha1           => "-1176510_29191",
        SecondarySaHa2           => "-1244857_28664",
        SecondarySaHa3           => "-1115761_73305",
        SecondaryLeak            => "-1259271_3472",
        SecondaryQtip            => "-1278110_-3460",
        SecondarySeal            => "-1341667_6309",
        SecondarySealQtip        => "-1356910_-7105",
        SecondaryUnknown1        => "-1381925_32122",
        SecondaryUnknown2        => "-1381924_32122",
        SecondaryMvTip0          => "-1444924_32122",
        SecondaryInc             => "-1644460_19460",
        SecondaryIncMid          => "-1883196_70395",
        SecondaryIncOtr          => "-1892555_60547",
        SecondaryExternal        => "-1793917_-61923",
        SecondaryLearn           => "-1900063_58623",
        SecondaryDump            => "-1981749_5626",
```

```
        # the day/month/year level so we default
        $year   = 12;
        $month  = 9;

        #just started a new day apparently, therefore increment $day
        $new_day && $day++;


        $event_time  = timelocal($sec,$min,$hour,$day,$month,2000+$year);
        $event_time .= $usec;

        return $event_time;
    }

    # Create a communication bridge with R (if you plan to use it)
    my $R = Statistics::R->new();

    # first arg is file to process
    $pcaptxt = shift;

    open(PCAPTXT,$pcaptxt) || die "Couldnt open $pcaptxt\n";

    print "\n\nProcessing $pcaptxt ... \n\n";

    while (<PCAPTXT>) {

        #extraneous characters at end of line ?
        chop;chop;

        # Beginning of command, Command; Cmd = Move Arm;
        if (/Command; cmd = (.*?);/) {

            $done       = 0;
            $doneDone   = 0;
            $diff_x     = 0;
            $diff_theta = 0;

            $op_name  = $1;
            $line_num = $.;

            #note if $in_cmd is set indicating an IN CMD after DONE
            #then clear it.
            if ($op_name eq "Move Arm" && $in_cmd) {
                push @cmdAfterDones, $.-13;
                $in_cmd = 0;
            }

        }

        # we do this in case pcap file starts with a response
        # instead of command.
        if (/Status; Cmd = (.*?);/) {
           $op_name = $1;
        }

        #command and timestamp
        #look for 09:58:14.114181  192.168.1.2 -> 192.168.1.1
        if (/(.*?)\s+192.168.1.1 -> 192.168.1.2/) {

           $cmd_time = parse_time($1,0);

        }

        #response and timestamp
        #look for 09:58:14.114181  192.168.1.1 -> 192.168.1.2
```

```
            $to_waypoint_name     = $to_x . "_" . $to_theta;

            $from_waypoint_name   = $from_x . "_" . $from_theta;
            $move_name            = $rev_stationsToCounts{$from_waypoint_name} .
                                          "--" . $rev_stationsToCounts{$to_waypoint_name
};
            $diff_x               = abs($final_x - $to_x);
            $diff_theta           = abs($final_theta - $to_theta);

            ($diff_x > 0) &&
                 print "Found a diff_x $diff_x $to_x $final_x $move_name at line $.\n";

            ($diff_theta > 0) &&
                 print "Found a diff_theta $diff_theta  $move_name at line $.\n";

            #create a $move_name hash key if it doesnt exit
            if (!exists($messages{$move_name} )) {
               $messages{$move_name} = create_record($op_name,$move_name);
               #print "$. - MOVE_NAME   $move_name $from_waypoint_name $to_waypoint_nam
e \n\n";

               #the @move_names will be used to iterate on later
               push @move_names, $move_name;

            }

            #push the $duration value onto the TIME_DIFFS array for this $move_name
            push @{$messages{$move_name}->{POS_DIFFS_X}},$diff_x;
            push @{$messages{$move_name}->{POS_DIFFS_THETA}},$diff_theta;

            #push the $duration value onto the TIME_DIFFS array for this $move_name
            push @{$messages{$move_name}->{TIME_DIFFS}}, $duration;

            #push the $line_num value onto the LINE_NUMS array for this $move_name
            push @{$messages{$move_name}->{LINE_NUMS}}, $line_num;

            #push the $cmd_time value onto the COMMAND_TIMES array for this $move_name
            push @{$messages{$move_name}->{COMMAND_TIMES}}, $cmd_time;

            #push the $rsp_time value onto the RESPONSE_TIMES array for this $move_name
            push @{$messages{$move_name}->{RESPONSE_TIMES}}, $rsp_time;

         }

      } #end of while PCAPTXT

    #now go through the hash and get at the Move Arm times for the various $move_names
    my ($loop_move_name);
    my ($num_commands);
    my ($min,$max,$mean,$stddev);


    sub footer {
      printf(" |
                             |\n");
      printf("  ====================================================================
========================== \n");
    }

format STDOUT=
# | @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< @<<<<< @<<<<< @<<<<< @###.##
@###.##    |
 | @<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< @<<<<< @#####.## @#####.## @###.
##   @###.##    |
$loop_move_name,$num_commands,$min,$max,$stddev,$mean
```

```
        my @ticks = 1..$num_commands;

        #towards GD::Graph object
        my @data;
        push (@data,\@ticks);
        push (@data,\@time_diffs);

        #push (@data,\@diffs_x);
        #push (@data,\@diffs_theta);
        #push (@data,\@pos_diffs);

        # setup legend labels
        #my @legend = qw( Command_To_Response_Time TimesEncoder_Count_Difference);

        my $graph  = GD::Graph::lines->new(800, 400);
        #my $graph = GD::Graph::linespoints->new(1000, 400);
        $graph->set(
             'title'            => "$pcaptxt $loop_move_name, SubSys Emulated $num_com
mands",
             'y_label'          => "Time in milliseconds",
             'x_label_skip'     => $num_commands/10,
             'y_number_format'  => '%2.2f',
             'x_number_format'  => '%d',
             #'show_values'     => 1,
             'y_max_value'      => (1.05 *$max),
             #'y_min_value'     => (.90*$mean),
             #'y_max_value'     => (1.10* $mean),
             'markers'          => '8',
             'marker_size'      => '1'
         );

        #set graph legend
        #$graph->set_legend(@legend);
        $graph->set_title_font(GD::gdSmallFont);

        open(IMG, ">${pcaptxt}_${loop_move_name}.png") or die $!;
        binmode IMG;
        print IMG $graph->plot(\@data)->png or die $graph->error;

        #this is for R plotting
        #if ($num_commands >1) {
             #Create an R plot for each command
             #my $output_file = "${pcaptxt}_${loop_move_name}.pdf";
             #my $title =  "$pcaptxt $loop_move_name";
             #my $x_ref = \@ticks;
             #my $y_ref = \@time_diffs;
             #$R->set('t', $title);
             #$R->set('x', $y_ref);
             #$R->send('d <- density(x)');
             #$R->run(qq`pdf("$output_file")`);
             #$R->send('plot(d,main=t)');
        #}

    }

    footer();

    #$R->run(q`dev.off()`);
    $R->stop();

    #print doneDones caught at line nums
    scalar(@doneDones)     && print "\nFound Done Dones at line numbers @doneDones\n";
    scalar(@cmdAfterDones) && print "\nFound InCmds After Dones at line numbers @cmdAf
terDones\n";
```